```
/*********************************************************************
*
*
*
*********************************************************************
* FileName:        LED7S.C
* Dependencies:    See INCLUDES section below
* Processor:       18F4520
* Compiler:        MCC18
* Company:         Jose Javier Gonzalez
*
* Author               Date        Comment
*~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
*********************************************************************/
#include <p18f4520.h>
#include  "LED7S.h"

/** V A R I A B L E S ********************************************************
    ****/

char DISPLAY[4]={0x00,0x00,0x00,0x00};

const char BCD7S[16] = {0b00111111,0b00000110,0b01011011,0b01001111,0
    b01100110,0b01101101,0b01111101,0b00000111,0b01111111,0b001101111,\
    //1-9 configs
                          0b01110111,0b01111100,0b00111001,0b01011110,0
                              b01111001,0b01110001}; //A-F configs
const char CC7S[4] = {0b00100000,0b00000010,0b00001000,0b00000100};
                                          //const char CCANDMASK = 0b
                                              110100001;
const char CCANDMASK=0b11010001;
char CCON=0;

/
    ********************************************************************
    ***************************
       Prototypes
*********************************************************************
    *************************/

/*********************************************************************
    Initialization of TRISA,TRISB,PORTA,PORTB,TMR0,INTCON
*********************************************************************/
void init_LED7S(void)
{
    TRISB=0x00;             //All PORTB as exits
    TRISA&=0b11010001;      //RA1-RA2-RA3-RA5 as exits
    PORTB=0x00;
    PORTA&=CCANDMASK;
    TRISC&=0b10111111;
    _hp=0;

    RCONbits.IPEN=1;
    INTCON=0b11100000;      //Global Interruts,Peripherical interrupts,
        external interrupts and TMR0 interrupt // Not PORTB change
        interrupt
                            //All Interrupt flags to 0
    INTCON2bits.TMR0IP=0;   //TMR0 Int priority to low
    T0CON=0b00011000;       //TMR OFF / 16 BIT mode // Fosc/4 / Rising
```

```
        Edge / PSA asigned / PSA to 128
    TMR0H= _TIMEH;
    TMR0L= _TIMEL;

}


/*************************************************************************
        NTO7S returns char configuration of a,b,c,d,e,f,g,dp of a char
            between 0-9 & A-F
*************************************************************************/
char NTO7S (char n){
    if(n<0||n>16)
        return 0x40; //returns horizontal line as an error
    return (char)BCD7S[n];
}

/*************************************************************************
   UPDATE : Functions updates DISPLAY vars with the new config, (FALSE:it
       is OVERLOADED to support multiple inputs)
_________________________________________________________________________
    ___________________________________
*************************************************************************/

    /*********************************************************
    UPDATE: 1 variable between 1-9999
    *********************************************************/
    void Update(int n){
        if(n>=0&&n<=9999){
            DISPLAY[0]=NTO7S(n%10);
            DISPLAY[1]=NTO7S((n/10)%10);
            DISPLAY[2]=NTO7S((n/100)%10);
            DISPLAY[3]=NTO7S((n/1000)%10);
        }
    }
    /*********************************************************
    UPDATE2: 2 variables between 1-99 each [ORDER= N1:N2]
    *********************************************************/
    void Update2(int n1,int n2){
        if(n2>=0&&n2<=99){
            DISPLAY[0]=(char)NTO7S(n2%10);
            DISPLAY[1]=(char)NTO7S(n2/10);
        }
        if(n1>=0&&n1<=99){
            DISPLAY[2]=(char)NTO7S(n1%10);
            DISPLAY[3]=(char)NTO7S(n1/10);
        }
    }
    /*********************************************************
    UPDATE4: 4 variables between 1-9 each [ORDER= N1:N2:N3:N4]
    *********************************************************/
    void Update4(char n1,char n2,char n3,char n4){
        if(n4>=0&&n4<=9)
            DISPLAY[0]=NTO7S(n4);
        if(n3>=0&&n3<=9)
            DISPLAY[1]=NTO7S(n3);
        if(n2>=0&&n2<=9)
            DISPLAY[2]=NTO7S(n2);
        if(n1>=0&&n1<=9)
```

```
            DISPLAY[3]=NTO7S(n1);
    }
    /*********************************************************
    UPDATE_L: Update Literal Value on the Nth display  ORDER:[CC3:CC2:CC1:
        CC0]
    *********************************************************/
    void Update_L(char config,char CC){
        if(CC>=0&&CC<=3)
            DISPLAY[CC]=config;
    }
    /*********************************************************
    UPDATE_H: Updates Hexadcimal Value on the display (0x0000-0xFFFF)
    *********************************************************/
    /*void Update_H(int h1){
        if(h1>=0x0000&&h1<=0xFFFF)
            Update_H2((char)h1>>8,(char)h1&0x00FF);
    }
    /*********************************************************
    UPDATE_H2: Updates 2 Hexadcimal Value on the display (0x00-0xFF) Each
        // 2 bytes represnted
    *********************************************************/
    void Update_H2(char h1,char h2){
        DISPLAY[0] = NTO7S(h2&0x0F);
        DISPLAY[1] = NTO7S((h2>>4)&0x0F);
        DISPLAY[2] = NTO7S(h1&0x0F);
        DISPLAY[3] = NTO7S((h1>>4)&0x0F);
    }
/*************************************************************************
_________________________________________________________________________
FINISH OF  UPDATE
*************************************************************************/




/*************************************************************************
        SET DECIMAL : Activates decimal point of the nth display    ORDER:
            [3:2:1:0]
*************************************************************************/
void Set_Decimal (char cc){
    if(cc>=0&&cc<=3)
        DISPLAY[cc]|=0b10000000;
}

/*************************************************************************
        ERASE DECIMAL : Deactivates decimal point of the nth display
            ORDER: [3:2:1:0]
*************************************************************************/
void Erase_Decimal (char cc){
    if(cc>=0&&cc<=3)
        DISPLAY[cc]&=0b01111111;
}
/*************************************************************************
        REFRESH : Turns on consecutiely all the display everytime the TMR0
            makes an interrupt
                : Resets TMR0 and Erase the flag
*************************************************************************/
void Refresh (void){
```

```
    TMR0H=_TIMEH;       // Siempre se debe escribir primero TMR0H y luego
        TMR0L
    TMR0L=_TIMEL;       // Recarga TMR0 LH
    _TMR0IF=0;      //INT Flag

                                //CCON is the Active Catode
    PORTA&=CCANDMASK;           //Erases previos catode(s)
    PORTA|=CC7S[CCON];          //Puts actual catode
    PORTB=DISPLAY[CCON++];      //Tunrs On the necesary LEDs
    CCON &=0x03;
}


/*

}



/*************************************************************************
                    EOF
*************************************************************************/
```